Performance Evaluation for SDN Deployment: an Approach based on Stochastic Network Calculus

A Scalable Method for Partitioning Workflows with Security Requirements over Federated Clouds

Zhenyu Wen
School of Computing Science
Newcastle University
Email: z.wen@newcastle.ac.uk

Jacek Cała
School of Computing Science
Newcastle University
Email: jacek.cala@newcastle.ac.uk

Paul Watson
School of Computing Science
Newcastle University
Email: paul.watson@newcastle.ac.uk

***Abstract*—The significant increase in the use of cloud computing, has led to an interest in partitioning applications over a set of public and private clouds in order to meet a range of non-functional requirements including performance (for example where private cloud resources alone are insufficient), dependability (e.g. to allow the application to continue to operate even if one cloud fails) and security (for example to ensure that sensitive data is restricted to sufficiently secure clouds and networks). This paper describes a novel deployment planning algorithm to partition complex workflow-based applications over federated clouds, while meeting security requirements. The security issues are based on our previous work which extends the Bell-LaPadula model to encompass cloud computing. Selecting the cheapest option for partitioning a workflow over a set of resources has been shown to be an NP-hard problem, which can take impractically long for partitioning large workflows over multiple clouds. We therefore introduce a novel adaptive partitioning algorithm to handle these large workflow applications, which significantly reduces the time required to choose a sufficiently-good partitioning option. This is based on generating an initial partitioning, and then adapting it to see if a better solution can be found by bringing together on the same node services with significant communication costs. The algorithm has been implemented and evaluated by using both randomly generated and real world scientific workflows. The experiment results show that our algorithm is thousands times quicker than the exhaustive algorithm presented in our previous work. Yet, on average it generates only 25% more costly solutions. We also compared this algorithm with two other methods commonly used to partition workflows over a set of clouds.**

***Index Terms*—cloud computing, workflow deployment, partitioning, scheduling, security**

I. INTRODUCTION

Cloud computing offers computing resources as a utility like water or electrical power, which means that any organisation or individual can easily rent remote computer and storage resources dynamically to the host networked applications. There are now several cloud providing companies, including Microsoft, Google and Amazon, and each company is making multiple, geographically distributed cloud data centres available. This enables customers to select an individual cloud data centre based on their requirements for price, functionality, latency and governance regulations. However, it also opens the possibility of running applications over a set of cloud data centres to meet availability requirements. This has led to an interest in the idea of federated clouds [1][2], but the main push for these has been security.

Some organisations have sensitive data or services that they are not prepared to host on public cloud. A potential solution to this also comes from the idea of federated clouds: deploy those parts of applications that are sensitive on trusted internal IT resources within an organisation (on what have come to be known as private clouds), but allow those parts with fewer security requirements to be deployed on public clouds where they can take advantage of their scalability and cost benefits.

Achieving this is not necessarily straightforward, especially for complex applications. There can be a very large number of options for how to deploy the data and services across a set of clouds, each of which will have a different cost model. Therefore, in this paper we tackle one of the main problems faced by organisations wishing to exploit federated clouds: how to select a deployment option that meets their security requirements, while at the same time minimising the cost that a given deployment will incur.

We describe the design and evaluation of an algorithm to partition an application structured as a workflow over a federated cloud in order to exploit the strengths of each cloud. The algorithm improves our previous method presented in [3], making it suitable for large workflow applications. The previous method, based on the Bell-LaPadula [4] Multi-Level Security model [5], gave a solution for partitioning a workflow over a set of clouds to meet certain security requirements. However, in order to find the cheapest workflow deployment, all of the potential partitioning options needed to be listed and ranked to find the cheapest one. Although this method gives the optimal and guarantees that the result is the cheapest deployment (we use ACO which means always cheapest option to represent this method in the following paragraphs), it is not very scalable. For example, it can take more than 15 minutes to optimally partition a workflow comprising 12 blocks (services and data items) over 4 clouds. The complexity of the method is $O(c^s)$, where c is the number of clouds and s is the number of blocks. Therefore increasing the number of the blocks in the workflow, raises the partitioning time exponentially.

Consequently, to handle more complex workflows and larger cloud federations our idea is to sacrifice some cost in order to reduce the time to produce a recommended deployment option. We present an approximate algorithm which still meets Bell-LaPadula security requirements. Its time complexity ap-

proximated closely to $O(2 \cdot s \cdot c)$ but gives an acceptable yet suboptimal result in terms of costs. We highlight the challenges and contributions of this paper as follows.

- The main contribution of this work is a deployment planning algorithm that minimises the time taken to derive a partitioning of a large workflow application across federated clouds so as to meet security requirements and reduce the price paid by the organisation for executing the workflow. The algorithm takes into account the three main sources of financial cost in the cloud: computation, data transfer and data storage. To estimate these costs we analysed the provenance traces of our case. Other work [6] [7] [8] describe how to predict those information using execution logs or workflow input data.
- We build on our previous work of using a version of the Bell-LaPadula Multi-level model easier to guarantee that the selected deployment option meets the organisation's security requirements.
- We evaluate our algorithm by using both randomly generated workflows, and a real world scientific workflow, and compare the experimental results with our previous work and other methods.

The paper is structured as follows. The following section reviews related work. The security problem and the cost issue are discussed in Section III. Next, the algorithm is presented in Section IV, followed by an illustrative example. In Section VI the complexity of the algorithm is briefly analysed. Finally, our experimental results are discussed and conclusion are drawn.

II. RELATED WORK

Cost optimisation has been a classic research topic for decades. Existing work has attempted to solve the workflow-mapping problem using DAG scheduling heuristics such as [9], [10], [11], to name just a few. However, these algorithms are all based on "free" grid resources and thus aim to minimise makespan. In clouds the providers charge a monetary cost depending on three factors (computation, data transfer and data storage). Our work takes these factors into account and partitions workflows over federated clouds and minimise the monetary cost.

To reduce the monetary cost various techniques has been proposed previously including rule-based techniques [12] and model-based techniques [13], [14]. In paper [14], the cost is minimised by transforming the structure of workflow applications in a single cloud. In contrast, our work uses a model-based technique on federated clouds.

Research related to federated clouds or multicloud environments is still new with little literature available. In [15], the authors introduce a pricing model and a truthful mechanism for scheduling workflow applications to different clouds. Chen and Deelman described in [16] how to use Genetic Algorithm (GA) to map a workflow to different clouds to minimise the monetary cost. While both of the papers are focused on minimising the monetary cost of completely executing workflow applications, neither consider the security issues that can limit the set of options.

Compared with our previous work [3], this algorithm avoids generating all possible results to achieve cost minimisation, which makes it much more efficient in dealing with more complex workflows (this is evaluated in SectionVII). In order to extend the evaluation, we have applied the security model to both GA and the Greedy Algorithm to compare the results with our algorithm. Our algorithm shows better performance in cost optimisation. The details are described in Section VII.

With regards to the security of cloud computing, most research focuses on improving security of the data centres [17], [18], [19] and [20], etc. [20] introduces a monitoring architecture to ensure the security and flexibility of a virtual machine. Conversely, we consider improving security by deploying services of a workflow application on different clouds to meet their security requirements.

In paper [21], the authors propose a approach to partition business workflow applications over a set of clouds to meet the security constraints. However, the paper focuses on minimising the communication to improve the reliability of the workflow enactment. In contrast, our aim is to minimise the monetary cost of enactment of scientific workflows which are often characterised by high demand in processing power and/or requiring the transfer and storage of large amounts of data.

III. PROBLEM DESCRIPTION

A workflow-based application consists of a set of services and data. It can be represented as a Directed Acyclic Graph (DAG), $G = (S, E)$, where the vertices S are the set of services, and edges E are a set of dependencies between those services. A workflow-based application can have several different types of dependency relationships, but here we only consider data dependency, which means a data item is generated from a source service and consumed by a destination service. For example, $e_{i,j}$ represents a data dependency between service s_i and service s_j. To represent data dependencies we use a distance matrix $D = [d_{i,j}]$ of size $|S| \times |S|$ where a $d_{i,j}$ value greater than zero indicates not only there is a dependency between s_i and s_j but also the size of data transmitted between them. Furthermore, C represents a set of clouds which are available for deployment.

A. Security Rules

A security model is needed to determine whether a deployment of services and data to a set of clouds meets an organisation's security requirements. We reuse our previous model [3], which builds on the Bell-LaPadula and incorporate the security levels of the clouds, data and services. Three security matrices represent these security level:

- $SecS$ is a $|S| \times 2$ matrix which describes the security levels of each service. Each service is assigned two security levels: "Clearance" and "Location". "Clearance" represents the service's highest security level, and "Location" is the required operation security level of the service in a specific application. The clearance of a service must be greater than or equal to its location. To make the rest

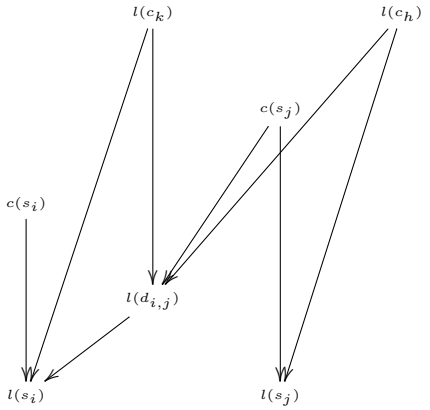

Fig. 1: The Security Level of Each Lattice

of the paper simpler, we split the $SecS$ into two $|S|$-dimensional vectors: CoS (clearance of service) and LoS (location of service).

- $SecD$ is a $|S| \times |S|$ distance matrix, where $SecD_{i,j}$ represents the security location of data item $d_{i,j}$.
- $SecC$ is a vector representing the security location of each cloud.

Our security model consists of three rules. Firstly, NRU or "no-read-up", means service s_j cannot read $d_{i,j}$, if the data's location is higher than its own clearance. Let CoS_j represents the clearance of service s_j; similarly, $SecD_{i,j}$ is $d_{i,j}$'s location value. Therefore, the NRU rule is defined as

$$NRU(d_{i,j}, s_j) = \begin{cases} true, & CoS_j \geq SecD_{i,j} \\ false, & \text{Otherwise} \end{cases}$$

Next, the "no-write-down" rule NWD indicates that service s_i cannot write data $d_{i,j}$ to lower location value than its own location. The location value of service s_i is LoS_i.

$$NWD(s_i, d_{i,j}) = \begin{cases} true, & SecD_{i,j} \geq LoSi \\ false, & \text{Otherwise} \end{cases}$$

Finally, when extending Bell-LaPdula to cloud computing, rule CS guarantees the security for the deployment of data and services on a cloud. It is defined such that location of a cloud must be greater than or equal to the location both any service and data that are hosted by the cloud. Note, that if data is transmitted from one cloud to another then this rule must be applied to both, the sending and receiving clouds.

$$CS(d^h_{i,j}, s^h_j) = \begin{cases} true, & SecC_{C_h} \geq LoS_i \\ & \text{and} \\ & SecC_{C_h} \geq SecD_{i,j} \\ false, & \text{Otherwise} \end{cases}$$

Figure 1 illustrates the relationship between the clearance and location attributes of services, data and clouds; where the "$\rightarrow$" represents the "$\geq$" relationship, and l, c are location and clearance respectively.

B. Representing Cost Requirements

We assume that the clouds are linked in a fully connected topology but the data can be freely transferred between clouds only if the security requirements described above are met. Additionally, a cloud can run several services at the same time. To represent cost we first define some basic metrics of our cost model:

- $Compu$ is a $|S| \times |C|$ matrix that represents computation cost such that $Compu_{i,h}$ is the cost of running service s_i on cloud c_h.
- The matrix Com represents a unit cost of data transmission from one cloud to another. For example, $Com_{h,f}$ means the cost of transferring 1GB of data from cloud c_h to c_f.
- $CStore$ is a vector for describing the cost of a unit data stored in a cloud for a unit time$CStore_h$, for instance, denotes the cost of stored 1GB of data for 1 hour on cloud c_h. Besides, this is only charged by source clouds when data cross cloud boundaries.
- The storage time of each unit of data is denoted in matrix $TStore$. For instance, $TStore_{i,j}$ is the storage time of $d_{i,j}$, which is equal to the sum of the execution time of services s_i and s_j and plus the data transfer time if crossing cloud boundaries.

Given these basic metrics we now define a set of cost functions that we will use later in our algorithm:

- First is the data storage cost:

$$SCOST(d^h_{i,j}) = d_{i,j} \times TStore_{i,j} \times CStore_h$$

  Where $d^h_{i,j}$ represents data $d_{i,j}$ stored on cloud c_h. Where data is transferred from cloud c_h to another cloud (to transfer it from one partition of the application to another held on a different cloud) we make the assumption that data only remains stored on the source cloud so as not to double-account for the cost. A reason for storing the output of a partition even after the data it generates has been sent to another cloud is that if the destination cloud fails, it provides a way to continue the computation on another cloud without having to restart the execution of the whole workflow.
- The communication cost of a set of data transferred from service s_i to s_j, which are deployed on cloud h and f respectively, can be defined as:

$$CCOST(s^h_i, s^f_j) = d_{i,j} \times Com_{h,f}$$

  Note that when both s_i and s_j are deployed on the same cloud the communication cost is 0.
- $SCOST$ and $CCOST$ are used to derive the key functions in our algorithm, SOC and COD. $SOC(s^f_i)$ is the costs incurred in bringing the input data consumed by service s_i to cloud c_f before the service is executed.

This includes the storage cost and communication cost

$$SOC(s_i^f) = \sum_{s_n^h \in Pred(s_i)} CCOST(s_n^h, s_i^f) + \sum_{s_n^h \in Pred(s_i)} SCOST(d_{n,i}^h)$$

Where $Pred(s_i)$ represents the set of immediate predecessor services that produce data consumed by service s_i. Furthermore, s_n^h indicates that one of s_i's predecessor service s_n was running on cloud c_h. If a service has no predecessors (i.e. it is one of the initial services in the workflow) then:

$$SOC(s_{entry}^h) = 0$$

The COD function denotes the extended cost of having a service deployed on a specific cloud. It is calculated by adding the computing cost of the service s_i to the transmission cost and storage cost of data sent from any of its predecessor services that are not in the same cloud.

$$COD(s_i^f) = Compu_{i,f} + SOC(s_i^f)$$

- We define a $|S| \times |C|$ matrix $PrePLAN = [preplan_{i,h}]$ which is used to determine the initial deployment of workflow services to clouds. Each value in the matrix is defined as $preplan_{i,h} = COD(s_i^h)$. Thus, each row of the matrix therefore contains COD values for a selected service as deployed on each specific cloud (except where deploying the service on a cloud would violate the security requirements). The initial deployment is then based on the smallest COD value in each row (i.e. for each service). Later, to improve this initial deployment we use the "Re-Deployment Method" (discussed in the following section) which generates the final deployment matrix for each service. The final deployment is stored in a $|S| \times |C|$ adjacency matrix DEP. Where $dep_{i,h} = 1$ indicates that service s_i is deployed on cloud c_h. Consequently, the total COST of deploying a workflow application is:

$$COST = \sum_{\substack{s_i \in S \\ \exists h \in C\ dep_{i,h}=1}} COD(s_i^h)$$

IV. ALGORITHM

The COD function is used to determine the initial deployment of services to clouds. It does not provide optimal deployment because it takes into account only the local costs related to each single service considered in isolation. Thus, the idea of our algorithm is to use more information in the planning of the final deployment and to make short term sacrifice for long term benefit. We call our algorithm NCF (not cheapest first). The algorithm consists of three steps. First, it starts by applying security rules to verify whether security requirements are met by the original workflow. We use rule CS to verify security for deploying services to clouds. Next, if the security requirements are met, we use COD to calculate the initial deployment matrix $PrePLAN$. Finally, the "Re-Deployment Method" is applied to improve the initial deployment, which works by combining services deployed in different clouds into a single cloud. The overall aim is to detect whether the costs can be reduced by avoiding intercloud communication and related storage costs.

The following sections describe the algorithm steps in more details.

A. Workflow Security Checking

The workflow is valid iff all return NRU and NWD values are true. OOtherwise, the workflow is invalid, security check returns error and the whole algorithm stops. The pseudocode of the "Workflow Security Check" is shown in Algorithm 1.

Algorithm 1 Workflow Security Check

```
D set of dependencies between related services
S set of service
Secure: =True
for d_{i,j} ∈ D do
  if not (NRU(d_{i,j}, s_j) and NWD(s_i, d_{i,j})) then
    Secure:=False
    Stop
  end if
end for
```

B. Initial Deployment

The initial deployment of services is based on the smallest COD value of each service taking into account security requirements checked by the CS rule. If a cloud c_h generates the smallest COD value but does not meet requirements imposed by CS, a cloud with the second smallest COD value is considered. The algorithm works until it finds a cloud that can meet the security requirements and the COD value associated with this cloud is stored in vector REC. If no cloud is found that meets CS, the algorithm stops. Algorithm 2 shows pseudocode for the initial deployment step.

Algorithm 2 Initial Deployment

```
S set of services in the workflow
REC init with INF
PrePLAN init with zero
for s_i ∈ S do
  for c_h ∈ Cloud do
    if CS == 1 then
      if COD < REC_i then
        REC_i = COD
        PrePLAN_{i,h} = 1
      end if
    end if
  end for
end for
```

C. Re-Deployment Method

The core idea behind the "Re-Deployment Method" is to avoid scheduling services to the clouds which bring huge communication cost. To realise this we use two functions "detect" and "replace" which are defined as follows:

$$detect(s_i) = \begin{cases} case1, & a \ \& \ \neg b \\ case2, & \neg a \ \& \ b \\ case3, & a \ \& \ b \\ case4, & \neg(a \ \& \ b) \end{cases}$$

$$s_{i,max} = max_{s_j \in Child(s_i)}(REC(s_j))$$
$$SETP(s_i) = Parent(s_{max}) \cup s_{max}$$
$$SETC(s_i) = Child(s_i) \cup s_i$$
$$MIN(SET) = min_{c_h \in C}(SET)$$

$$\sum_{s_h \in SETP(s_i)} REC(s_h) > MIN(SETP(s_i)) \qquad \text{a}$$
$$\sum_{s_h \in SETC(s_i)} REC(s_h) > MIN(SETC(s_i)) \qquad \text{b}$$

From the description above, the minimal COD value of each service and the pre-planned deployment are recorded in REC and $PrePLAN$ respectively. The $detect$ function determines four different cases based on this and additional information:

- $s_{i,max}$ denotes the service s_i's child service with maximum COD value.
- $SETP(s_i)$ is a set which includes service $s_{i,max}$ and all its parent services.
- $SETC(s_i)$ includes s_i and all its child services.
- $MIN(SET)$ is the minimum cost of deploying the services from SET on a single cloud which meets security requirements of all services in SET.
- a is true if the cost of the initial deployment of the services in $SETP(s_i)$ is greater than the cost of $MIN(SETP(s_i))$
- b is true if the cost of the initial deployment of the services in $SETC(s_i)$ is greater than the cost of $MIN(SETC(s_i))$

Given all this information $detect$ returns four different sets of services. In case1 it returns $SETP(s_i)$, in case2 it returns $SETC(s_i)$, in case3 it returns one of $SETP$, $SETC$ which has smaller MIN value. Finally, in case4 it returns $\{s_i\}$.

After the services are selected by $detect$, the $replace$ function is invoked to assign these services into a cloud which minimises deployment cost. The pseudocode is shown in Algorithm 3.

Algorithm 3 Re-Deployment Method

```
US = S // is a set of unscheduled services
for s_i ∈ US do
  switch(detect(s_i))
  case1:replace(SETP(s_i))
  case2:replace(SETC(s_i))
  case3:
  if MIN(SETP) > MIN(SETC) then
    replace(SETC(s_i))
  else
    replace(SETP(s_i))
  end if
  case4: replace({s_i})
end for
```

V. An Illustrative Example

The workflow, shown in Figure 2, will be used to demonstrate the algorithm. More complicated workflows will be evaluated in section VII.

Workflow and cloud cost information are shown in Tables I, II, III, IV. Additionally, the security levels of the services, data and clouds are also required, which are indicated in Table V.

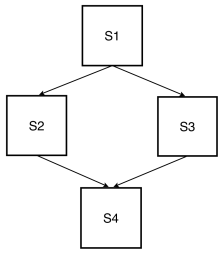

Fig. 2: Sample Workflow Application

Before assigning the services to clouds, the security of the workflow is checked. From Table V, every service and data meet the security rules NRU and NWD.

Next, the workflow has to be pre-assigned by following the function COD and security rule CS. The initial deployment is shown in Figure 3 and the cost is 1045.

To improve on the initial deployment, the "Re-Deployment Method" is applied. When $detect(s1)$ is applied, the value of $\sum_{s_h \in SETP(s_i)} REC(s_1) = 530$ and $MIN(SETP(s1)) = 300$ ($SETP(s1)$ includes $s1$ and $s3$, and " Cloud1" is the one to minimise the cost). In addition, $\sum_{s_h \in SETC(s_i)} REC(s_1) = 845$, while $MIN(SETC(s1)) = 550$ ($s1$, $s2$ and $s3$ are selected for $SETC(s1)$, and "Cloud1" is also the best choice). In this example, both case1 and case2 are satisfied. However, the former case has cheaper MIN value, therefore assigning services in $SETP(s1)$ to "Cloud1".

Next, after applying the $detect$ method to "s2", "s2" belongs case4, and is assigned to "Cloud1" after $replace(s2)$ is invoked. Similarly, service "s4" is also allocated to "Cloud1" after the application of "Re-Deployment Method". Conse-

Matrix	Element	Clearance	Location
$SecS$	S1	0	0
	S2	1	1
	S3	1	1
	S4	1	0
$SecC$	C0		0
	C1		1
$SecD$	$d_{s1,s2}$		0
	$d_{s1,s3}$		0
	$d_{s2,s4}$		1
	$d_{s3,s4}$		1

TABLE V: The Security Levels for Services, Data and Clouds

Service	C0	C1
$S1$	50	100
$S2$	100	200
$S3$	150	250
$S4$	160	200

TABLE I: The CPU Cost in Clouds

Cloud	C0	C1
C0	0	10
C1	20	0

TABLE II: Cloud Communication Costs (per GB)

Data	Time (hour)	Size(GB)
$d_{s1,s2}$	15	10
$d_{s1,s3}$	15	20
$d_{s2,s4}$	5	8
$d_{s3,s4}$	5	6

TABLE III: Data Size and Data Storage Time

Cloud	Cost
C0	0.1
C1	0.2

TABLE IV: Cloud Storage Cost of Clouds (per GB Hour)

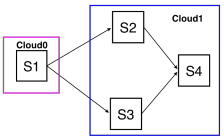

Fig. 3: Pre-Deployment Workflow System

quently, all services are assigned to "Cloud1", and the cost is 750.

VI. COMPLEXITY ANALYSIS

In our algorithm, we split the workflow security check and pre-planned deployment into two parts, because this can make the algorithm easier to understand. However, they can be combined in one step. This makes calculations more efficient and results in complexity $O(|E| \times |C|)$, where E is the set of data dependency edges and C is the set of clouds. The $detect$ and $replace$ functions have to compare the cost of a service and its immediate predecessor services in a pre-planned situation with the cost of deploying these services in one cloud. However, the complexity of the algorithm is also impacted by the structure of the workflow. If the workflow is linear, the complexity in the worst case becomes $O(|E + S| \times |C|)$. Conversely, for a star-shaped workflow the best case complexity is $O(|E| \times |C|)$.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we compare and evaluate our algorithm with ACO and other popular algorithms. Two types of workflow are tested in our experiments. One is randomly generated, the other is an example of a real world scientific workflow.

A. Randomly Generated Workflows

In this case, the workflow is represented by a distance matrix D in which if $d_{i,j}$ is greater than 0 that means service s_i sends data to service s_j. The value represents the size of data that is transferred between them. We create a $n \times n$ matrix with "-1" as the initial value. Next, the values of the matrix are assigned randomly, however we make sure that workflows are connected and acyclic.

In the experiment, we consider five different clouds that are assigned security levels from 0 to 4 randomly and stored in SecC vector. The other required matrices are generated in a similar way.

The experiments are implemented in the Java language and run on a 4-Cores machine with 2 GHz Intel Core i7 processor, 8G RAM and OS X MAvericks.

Figure 4 denotes the execution time of the NCF algorithm for different workflows which have different numbers of blocks, as shown on the x-axis from 2 to 30. In all cases the time was less than 1 millisecond. However, the execution time does not grow linearly with the number of blocks because the time complexity depends also on the structure of the workflow. In our experiment we can control the number of clouds, services and edges but the structure of the workflow is randomly generated. Note also that due to huge differences in the execution time of NCF and other algorithms, we present in the Figures ratio alg/NCF rather than absolute execution time.

The time cost of each column is the mean value of ten different structures of workflows. For example, we calculate the time cost of a workflow with five blocks by mean of the time cost of the algorithm which is applied to ten different 5 blocks workflows. As Figure 5 shows, the time cost of the ACO is significant ascent when the workflow which has more than 10 blocks. Figure 5 compares the ACO and NCF algorithms. The x-axis shows the number of blocks in random workflows, which is from 2 to 12 blocks, and the y-axis is the ratio of execution time of ACO by execution time of NCF. As presented there is a sudden surge in run time of the ACO algorithm when workflows have more than 10 blocks

The relative monetary cost of both algorithms is displayed in Figure 6. In most cases NCF is very close to the optimum with the worst case result we found being 25% higher than reported by ACO. The reason for this gap is that NCF can only improve the result by adjusting the deployment of the immediate predecessor services of the current planning service. However, further away services are not considered by our algorithm.

To generate the results of more complex workflows by using ACO is very time costly; therefore we also compare our algorithm with GA and Greedy Algorithm (GR). Both are popular methods to deploy large workflow systems on federated clouds proposed in [16]. As Figure 7 shows, the deployment options generated by our algorithm are closer to the optimal options. For 30-block workflow NCF generates solution that is 20% less costly than GA and 36% less costly than GR Besides, NCF is thousands of times faster than GA to generate a deployment option, and only 13 times slower than GR in the worst case, as shown in Figure 8, 9. These experimental results are generated by testing more than 750 workflows, between 5 blocks and 30 blocks.

B. Workflows from a Real Scientific Application

To verify our algorithm for real workflow, we used one from the Cloud e-Genome project[22] (Figure 10). The project's overall goal is to facilitate the adoption of genetic testing in clinical practice at a population scale. To realise this goal, Cloud e-Genome uses workflow modelling to program the whole exome sequencing pipeline, cloud computing to run

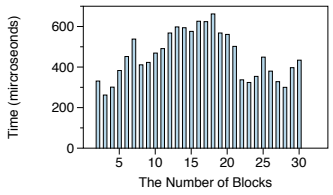

Fig. 4: Execution Time of NCF

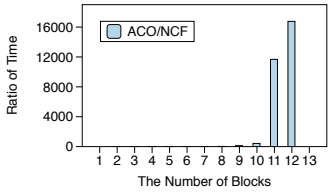

Fig. 5: Time Ratio between ACO and NCF

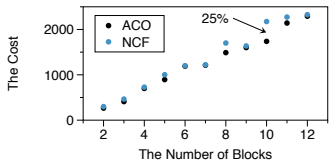

Fig. 6: The Cost of ACO and NCF

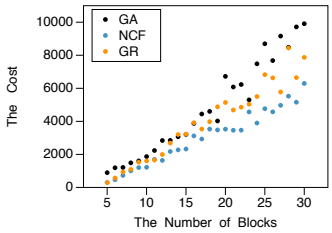

Fig. 7: The Cost of NCF, GA and Greedy

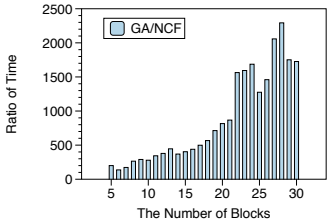

Fig. 8: Time Ratio between GA and NCF

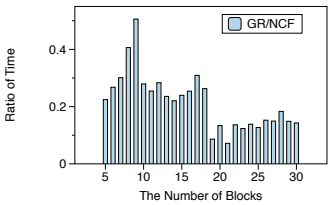

Fig. 9: The Cost of Greedy and NCF

Cloud	Pu1	Pr1	Pu2	Pr2
Security	0	1	0	1
CPU	1.68(/hour)	3.41(/hour)	1.40(/hour)	3.23(/hour)
EC2	0	0.08	0.02	0.07
EC(private)	0.08	0	0.07	0.12
Azure	0.06	0.11	0	0.1
Azure(private)	0.11	0.16	0.1	0

TABLE VII: Cloud Cost (U.S. Dollar per GB)

the workflows on large scale and provenance of workflow enactment to achieve reproducibility. All these aspects are supported by the e-Science Central platform (e-Science Central) [23] used to develop the pipeline

Although security aspects are not in the central focus of this pilot project, guaranteeing that human genome data can be securely processed on the cloud is a key issue. Therefore, we selected a workflow from Cloud e-Genome and modelled its security requirements by assigning security levels as shown in Tables VIa and VIb. The services "Data In" and "Data Out" are e-Science Central storage services which are not represented by blocks. Therefore they were not included in the Figure 10

Using logs collected by e-Science Central, we were able to determine the size of data transferred between workflow blocks execution times. Table VIa includes execution times in hours; 0 means that the times was less than 1 minute. Table VIb shows data sizes in GB and 0 means the size was less than 1 MB. From this data we calculated the cost of running the workflow in clouds offered by two major cloud providers.

The pricing of the clouds is shown in Table VII. We chose the same type of Virtual Machines in both clouds and also refer to public and private cloud setup offered by both providers; the private cloud setup is more secure yet more expensive. In the same table we present also data transfer costs between the clouds.

From these inputs, services $S1$ and $S5$ are assigned to Azure public cloud, and others are located in Azure private cloud. The total cost is 84.319 dollars.

VIII. Conclusion

This paper has described a novel, efficient scalable algorithm to automatically partition complex workflows over a federated clouds while meeting security requirements and minimizing exection cost. The main contribution of this paper is to redesign the exact algorithm presented in our previous work to enable it to be applied in real world scenarios by reducing the time it takes to generate a low-cost, secure partitioning option.

The algorithm was tested on randomly generated workflows and real world scientific workflows. Comparing with other methods, the time complexity of our algorithm has a significant advantage, and the cost is very close to the cost of the optimal deployments and less than the cost of deployments that generated from widely used algorithms. In the future work we will extend the algorithm to handle dynamic changes in the cloud environment (such as cloud failure) and will develop a tool to real-time monitoring of clouds and workflows.

References

[1] Z. Wen and P. Watson, "Dynamic exception handling for partitioned workflow on federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec 2013, pp. 198–205.

[2] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, July 2010, pp. 123–130.

[3] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 180–188.

[4] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations," MITRE Corporation, Tech. Rep., Mar. 1973.

[5] C.E.Landwehr, "Formal models for computer security," *ACM Computing Surveys*, vol. 13, 1981.

[6] M. Dobber, R. van der Mei, and G. Koole, "Effective prediction of job processing times in a large-scale grid environment," in *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*, 2006, pp. 359–360.

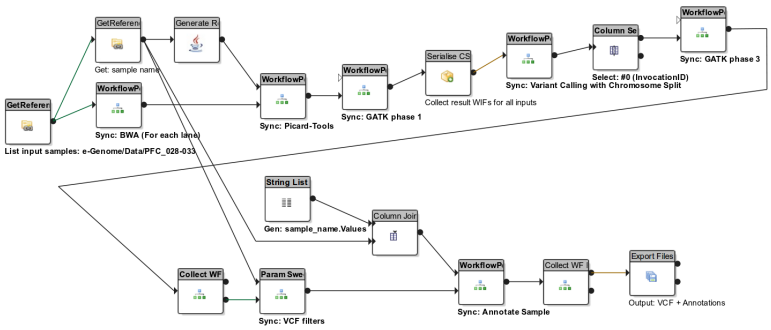

Fig. 10: e-Genome Workflow

Service	Name	Clarence	Location	Execution Time(hours)
Data In	S1	0	0	0
List Samples	S2	0	0	0
Get Samples Info	S3	0	0	0
Generate MetaData	S4	0	0	0
Align Sequences	S5	0	0	6.42
Clean Sequences	S6	0	0	4.05
Recalibareate Sequences	S7	0	0	11.45
Serialise CSV	S8	0	0	0
Export File	S9	0	0	0
Detect Variants	S10	1	1	4.46
Column Select	S11	0	0	0
CSVExport	S12	0	0	0
Recalibareate Variants	S13	1	1	1.425
Filter Variants	S14	1	1	0.5
Generate String List	S15	0	0	0
Column Join	S16	0	0	0
Annotate Sample	S17	1	1	0.86
Export File	S18	1	1	0
Data Out	S19	1	1	0

(a) Services Representation and Security and Execution Time

Service	Location	Size (GB)
$S_{1,2}$	0	0
$S_{1,3}$	0	0
$S_{1,5}$	0	24.99
$S_{2,3}$	0	0
$S_{2,5}$	0	0
$S_{2,15}$	0	0
$S_{2,17}$	0	0
$S_{3,4}$	0	0
$S_{4,6}$	0	0
$S_{5,6}$	0	18.63
$S_{6,7}$	1	15.56
$S_{7,8}$	0	0
$S_{7,10}$	1	6.87
$S_{8,9}$	0	0
$S_{8,10}$	0	0
$S_{9,19}$	0	0
$S_{10,11}$	0	0
$S_{10,13}$	1	0.074
$S_{11,12}$	0	0
$S_{11,13}$	0	0
$S_{12,19}$	0	0
$S_{13,14}$	1	0.09
$S_{14,17}$	1	0.055
$S_{15,16}$	0	0
$S_{16,17}$	0	0.11
$S_{17,18}$	1	0
$S_{18,19}$	1	0.11

(b) Data Security

TABLE VI: Workflow Security

[7] T. Miu and P. Missier, "Predicting the execution time of workflow activities based on their input features," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, Nov 2012, pp. 64–72.

[8] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer, "A hybrid intelligent method for performance modeling and prediction of workflow activities in grids," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, May 2009, pp. 339–347.

[9] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/71.993206

[10] I. Ahmad and Y.-K. Kwok, "A new approach to scheduling parallel programs using task duplication," in *Parallel Processing, 1994. Vol. 1. ICPP 1994. International Conference on*, vol. 2, Aug 1994, pp. 47–51.

[11] B. Kruatrachue and T. Lewis, "Grain size determination for parallel processing," *Software, IEEE*, vol. 5, no. 1, pp. 23–32, Jan 1988.

[12] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, Nov 2012, pp. 1–11.

[13] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1011 – 1026, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000744

[14] A. C. Zhou and B. He, "A. c. zhou and b. he," *IEEE Transactions on Cloud Computing*, vol. pp, no. 99, pp. 1–1, 2014.

[15] H. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1203–1212, June 2013.

[16] W. Chen and E. Deelman, "Integration of workflow partitioning and resource provisioning," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 764–768.

[17] E. Grosse, J. Howie, J. Ransome, J. Reavis, and S. Schmidt, "Cloud computing roundtable," pp. 17–23, 2010.

[18] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 693–702.

[19] C. Li, A. Raghunathan, and N. Jha, "Secure virtual machine execution under an untrusted management os," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, July 2010, pp. 172–179.

[20] B. Payne, M. de Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, Dec 2007, pp. 385–397.

[21] E. Goettelmann, W. Fdhila, and C. Godart, "Partitioning and cloud deployment of composite web services under security constraints," in *IEEE International Conference on Cloud Engineering (IC2E)*. San Francisco, California, USA: IEEE Computer Society, March 2013. [Online]. Available: http://eprints.cs.univie.ac.at/3565/

[22] J. Cala, Y. X. Xu, E. A. Wijaya, and P. Missier, "From scripted HPC-based NGS pipelines to workflows on the cloud," in *Procs. C4Bio workshop, co-located with the 2014 CCGrid conference*. Chicago, IL: IEEE, 2014.

[23] W. P. Hiden H, Woodman S and C. J, "Developing cloud applications using the e-science central platform," *Royal Society of London. Philosophical Transactions A. Mathematical, Physical and Engineering Sciences*, vol. 371, p. 20120085, 2013.